INTERNATIONAL STANDARD

ISO/IEC 14496-10

Third edition
2005-12-15

Corrected version
2006-03-01

AMENDMENT 2
2007-12-01

Information technology — Coding of audio-visual objects

Part 10: **Advanced Video Coding**

AMENDMENT 2: New profiles for professional applications

Technologies de l'information — Codage des objets audiovisuels

Partie 10: Codage visuel avancé

AMENDEMENT 2: Nouveaux profils pour applications professionnelles

Reference number
ISO/IEC 14496-10:2005/Amd.2:2007(E)

Foreword

ISO (the International Organization for Standardization) and IEC (the International Electrotechnical Commission) form the specialized system for worldwide standardization. National bodies that are members of ISO or IEC participate in the development of International Standards through technical committees established by the respective organization to deal with particular fields of technical activity. ISO and IEC technical committees collaborate in fields of mutual interest. Other international organizations, governmental and non-governmental, in liaison with ISO and IEC, also take part in the work. In the field of information technology, ISO and IEC have established a joint technical committee, ISO/IEC JTC 1.

International Standards are drafted in accordance with the rules given in the ISO/IEC Directives, Part 2.

The main task of the joint technical committee is to prepare International Standards. Draft International Standards adopted by the joint technical committee are circulated to national bodies for voting. Publication as an International Standard requires approval by at least 75 % of the national bodies casting a vote.

Attention is drawn to the possibility that some of the elements of this document may be the subject of patent rights. ISO and IEC shall not be held responsible for identifying any or all such patent rights.

Amendment 2 to ISO/IEC 14496-10:2005 was prepared by Joint Technical Committee ISO/IEC JTC 1, *Information technology*, Subcommittee SC 29, *Coding of audio, picture, multimedia and hypermedia information*.

Amendment 2 to ISO/IEC 14496-10:2005 is technically aligned with ITU-T Rec. H.264 (2005)/Amd.2 (2007), but is not published as identical text.

Information technology — Coding of audio-visual objects

Part 10: **Advanced Video Coding**

AMENDMENT 2: New profiles for professional applications

In subclause 0.4, replace the following:

ITU-T Rec. H.264 | ISO/IEC 14496-10 version 1 refers to the first (2003) approved version of this Recommendation | International Standard.

ITU-T Rec. H.264 | ISO/IEC 14496-10 version 2 refers to the integrated text containing the corrections specified in the first technical corrigendum.

ITU-T Rec. H.264 | ISO/IEC 14496-10 version 3 refers to the integrated text containing both the first technical corrigendum (2004) and the first amendment, which is referred to as the "Fidelity range extensions".

ITU-T Rec. H.264 | ISO/IEC 14496-10 version 4 (the current specification) refers to the integrated text containing the first technical corrigendum (2004), the first amendment (the "Fidelity range extensions"), and an additional technical corrigendum (2005). In the ITU-T, the next published version after version 2 was version 4 (due to the completion of the drafting work for version 4 prior to the approval opportunity for a final version 3 text).

with:

ITU-T Rec. H.264 | ISO/IEC 14496-10 version 1 refers to the first (2003) approved version of this Recommendation | International Standard. The first published version in ISO/IEC corresponded to version 1.

ITU-T Rec. H.264 | ISO/IEC 14496-10 version 2 refers to the integrated text containing the corrections specified in the first technical corrigendum. The first fully-published version in the ITU-T was version 2, due to the development of the corrigendum during the publication process. Version 2 was also published in integrated form by ISO/IEC.

ITU-T Rec. H.264 | ISO/IEC 14496-10 version 3 refers to the integrated text containing both the first technical corrigendum (2004) and the first amendment, which is referred to as the "Fidelity range extensions".

ITU-T Rec. H.264 | ISO/IEC 14496-10 version 4 refers to the integrated text containing the first technical corrigendum (2004), the first amendment (the "Fidelity range extensions"), and an additional technical corrigendum (2005). In both ITU-T and ISO/IEC, the next complete published version after version 2 was version 4.

ITU-T Rec. H.264 | ISO/IEC 14496-10 version 5 refers to the integrated version 4 text with its specification of the High 4:4:4 profile removed. In ISO/IEC, the changes from version 4 to version 5 were published as a corrigendum text.

ITU-T Rec. H.264 | ISO/IEC 14496-10 version 6 refers to the integrated version 5 text after its amendment to support additional colour space indicators. In the ITU-T, the changes for versions 5 and 6 were approved and published as a single amendment.

ITU-T Rec. H.264 | ISO/IEC 14496-10 version 7 refers to the integrated version 6 text after its amendment to define five new profiles intended primarily for professional applications (the High 10 Intra, High 4:2:2 Intra, High 4:4:4 Intra, CAVLC 4:4:4 Intra, and High 4:4:4 Predictive profiles) and two new types of supplemental enhancement information (SEI) messages (the post-filter hint SEI message and the tone mapping information SEI message).

In the second paragraph of subclause 0.6, replace the sentence:

With the exception of the transform bypass mode of operation for lossless coding in the I_PCM mode of operation in all profiles, the algorithm is typically not lossless, as the exact source sample values are typically not preserved through the encoding and decoding processes.

with:

With the exception of the transform bypass mode of operation for lossless coding in the High 4:4:4 Intra, CAVLC 4:4:4 Intra, and High 4:4:4 Predictive profiles, and the I_PCM mode of operation in all profiles, the algorithm is typically not lossless, as the exact source sample values are typically not preserved through the encoding and decoding processes.

In the third paragraph of subclause 0.7, replace the sentence:

Annex A specifies six profiles (Baseline, Main, Extended, High, High 10 and High 4:2:2), each being tailored to certain application domains, and defines the so-called levels of the profiles.

with:

Annex A specifies eleven profiles (Baseline, Main, Extended, High, High 10, High 4:2:2, High 4:4:4 Predictive, High 10 Intra, High 4:2:2 Intra, High 4:4:4 Intra, and CAVLC 4:4:4 Intra), each being tailored to certain application domains, and defines the so-called levels of the profiles.

Replace subclause 3.6 with the following:

3.6 **arbitrary slice order**: A *decoding order* of *slices* in which the *macroblock address* of the first *macroblock* of some *slice* of a *picture* may be less than the *macroblock address* of the first *macroblock* of some other preceding *slice* of the same *coded picture* or, in the case of a *picture* that is coded using three separate colour planes, some other preceding *slice* of the same colour plane.

Replace subclause 3.75 with the following:

3.75 **macroblock**: A 16x16 *block* of *luma* samples and two corresponding *blocks* of *chroma* samples of a *picture* that has three sample arrays; or a 16x16 *block* of samples of a monochrome *picture* or a *picture* that is coded using three separate colour planes. The division of a *slice* or a *macroblock pair* into macroblocks is a *partitioning*.

Replace subclause 3.136 with the following:

3.136 **slice**: An integer number of *macroblocks* or *macroblock pairs* ordered consecutively in the *raster scan* within a particular *slice group*. For the *primary coded picture*, the division of each *slice group* into slices is a *partitioning*. Although a slice contains *macroblocks* or *macroblock pairs* that are consecutive in the *raster scan* within a *slice group*, these *macroblocks* or *macroblock pairs* are not necessarily consecutive in the *raster scan* within the *picture*. The addresses of the *macroblocks* are derived from the address of the first *macroblock* in a slice (as represented in the *slice header*), the *macroblock to slice group map*, and, when a *picture* is coded using three separate colour planes, a colour plane identifier.

In subclause 6.2 "Source, decoded, and output picture formats", make the following changes.

Replace the following:

The variables SubWidthC, and SubHeightC are specified in Table 6-1, depending on the chroma format sampling structure, which is specified through chroma_format_idc. An entry marked as "-" in Table 6-1 denotes an undefined value for SubWidthC or SubHeightC. Other values of chroma_format_idc, SubWidthC, and SubHeightC may be specified in the future by ITU-T | ISO/IEC.

Table 6-1 –SubWidthC, and SubHeightC values derived from chroma_format_idc

chroma_format_idc	Chroma Format	SubWidthC	SubHeightC
0	monochrome	-	-
1	4:2:0	2	2
2	4:2:2	2	1
3	4:4:4	1	1

with:

The variables SubWidthC and SubHeightC are specified in Table 6-1, depending on the chroma format sampling structure, which is specified through chroma_format_idc and separate_colour_plane_flag. An entry marked as "-" in Table 6-1 denotes an undefined value for SubWidthC or SubHeightC. Other values of chroma_format_idc, SubWidthC, and SubHeightC may be specified in the future by ITU-T | ISO/IEC.

Table 6-1 –SubWidthC, and SubHeightC values derived from chroma_format_idc and separate_colour_plane_flag

chroma_format_idc	separate_colour_plane_flag	Chroma Format	SubWidthC	SubHeightC
0	0	monochrome	-	-
1	0	4:2:0	2	2
2	0	4:2:2	2	1
3	0	4:4:4	1	1
3	1	4:4:4	-	-

Replace the following paragraph:

In 4:4:4 sampling, each of the two chroma arrays has the same height and width as the luma array.

with:

In 4:4:4 sampling, depending on the value of separate_colour_plane_flag, the following applies.

– If separate_colour_plane_flag is equal to 0, each of the two chroma arrays has the same height and width as the luma array.

– Otherwise (separate_colour_plane_flag is equal to 1), the three colour planes are separately processed as monochrome sampled pictures.

Replace the following paragraph:

The number of bits necessary for the representation of each of the samples in the luma and chroma arrays in a video sequence is in the range of 8 to 12, and the number of bits used in the luma array may differ from the number of bits used in the chroma arrays.

with:

The number of bits necessary for the representation of each of the samples in the luma and chroma arrays in a video sequence is in the range of 8 to 14, and the number of bits used in the luma array may differ from the number of bits used in the chroma arrays.

Replace the following paragraph:

– If chroma_format_idc is equal to 0 (monochrome), MbWidthC and MbHeightC are both equal to 0 (as no chroma arrays are specified for monochrome video).

with:

– If chroma_format_idc is equal to 0 (monochrome) or separate_colour_plane_flag is equal to 1, MbWidthC and MbHeightC are both set equal to 0.

In subclause 6.3 "Spatial subdivision of pictures and slices", make the following changes:

Replace the following:

Each macroblock is comprised of one 16x16 luma array and, when the video format is not monochrome, two corresponding chroma sample arrays. When macroblock-adaptive frame/field decoding is not in use, each macroblock represents a spatial rectangular region of the picture. For example, a picture may be divided into two slices as shown in Figure 6-7.

with:

Each macroblock is comprised of one 16x16 luma array and, when the chroma sampling format is not equal to 4:0:0 and separate_colour_plane_flag is equal to 0, two corresponding chroma sample arrays. When separate_colour_plane_flag is equal to 1, each macroblock is comprised of one 16x16 luma or chroma sample array. When macroblock-adaptive frame/field decoding is not in use, each macroblock represents a spatial rectangular region of the picture. For example, a picture may be divided into two slices as shown in Figure 6-7.

When a picture is coded using three separate colour planes (separate_colour_plane_flag is equal to 1), a slice contains only macroblocks of one colour component being identified by the corresponding value of colour_plane_id, and each colour component array of a picture consists of slices having the same colour_plane_id value. Coded slices with different values of colour_plane_id within an access unit can be interleaved with each other under the constraint that for each value of colour_plane_id, the coded slice NAL units with that value colour_plane_id shall be in the order of increasing macroblock address for the first macroblock of each coded slice NAL unit.

Insert a new subclause 6.4.3.1 as follows:

6.4.3.1 Inverse 4x4 Cb or Cr block scanning process for ChromaArrayType equal to 3

This process is only invoked when ChromaArrayType is equal to 3.

The inverse 4x4 chroma block scanning process is identical to inverse 4x4 luma block scanning process as specified in subclause 6.4.3 when substituting the term "luma" with the term "Cb" or the term "Cr", and substituting the term "luma4x4BlkIdx" with the term "cb4x4BlkIdx" or the term "cr4x4BlkIdx" in all places in subclause 6.4.3.

Insert a new subclause 6.4.4.1 as follows:

6.4.4.1 Inverse 8x8 Cb or Cr block scanning process for ChromaArrayType equal to 3

This process is only invoked when ChromaArrayType is equal to 3.

The inverse 8x8 chroma block scanning process is identical to inverse 8x8 luma block scanning process as specified in subclause 6.4.4 when substituting the term "luma" with the term "Cb" or the term "Cr", and substituting the term "luma8x8BlkIdx" with the term "cb8x8BlkIdx" or the term "cr8x8BlkIdx" in all places in subclause 6.4.4.

In subclause 6.4.8 "Derivation processes for neighbouring macroblocks, blocks, and partitions", make the following changes.

Insert the following sentence after the paragraph starting with "Subclause 6.4.8.2 specifies*":*

Subclause 6.4.8.2.1 specifies the derivation process for neighbouring 8x8 chroma blocks for ChromaArrayType equal to 3.

Insert the following sentence after the paragraph starting with "Subclause 6.4.8.4 specifies".

Subclause 6.4.8.4.1 specifies the derivation process for neighbouring 4x4 chroma blocks for ChromaArrayType equal to 3.

Replace the paragraph starting with "Table 6-2 specifies*" with the following:*

Table 6-2 specifies the values for the difference of luma location (xD, yD) for the input and the replacement for N in mbAddrN, mbPartIdxN, subMbPartIdxN, luma8x8BlkIdxN, cb8x8BlkIdxN, cr8x8BlkIdxN, luma4x4BlkIdxN, cb4x4BlkIdxN, cr4x4BlkIdxN, and chroma4x4BlkIdxN for the output. These input and output assignments are used in subclauses 6.4.8.1 to 6.4.8.5. The variable predPartWidth is specified when Table 6-2 is referred to.

Insert a new subclause 6.4.8.2.1 as follows:

6.4.8.2.1 Derivation process for neighbouring 8x8 chroma blocks for ChromaArrayType equal to 3

This process is only invoked when ChromaArrayType is equal to 3.

The derivation process for neighbouring 8x8 chroma block is identical to the derivation process for neighbouring 8x8 luma block as specified in subclause 6.4.8.2 when substituting the term "luma" with the term "Cb" or the term "Cr", and substituting the term "luma8x8BlkIdx" with the term "cb8x8BlkIdx" or the term "cr8x8BlkIdx" in all places in subclause 6.4.8.2.

In subclause 6.4.8.4 "Derivation process for neighbouring 4x4 chroma blocks", make the following changes.

*Replace the paragraph starting with "*Depending on chroma_format_idc*" with the following:*

The position (x, y) of the upper-left sample of the 4x4 chroma block with index chroma4x4BlkIdx is derived by

x = InverseRasterScan(chroma4x4BlkIdx, 4, 4, 8, 0) (6-25)
y = InverseRasterScan(chroma4x4BlkIdx, 4, 4, 8, 1) (6-26)

Update all equation numbers in the subsequent subclauses due to the removal of equation 6-27 and 6-28.

Insert a new subclause 6.4.8.4.1 as follows:

6.4.8.4.1 Derivation process for neighbouring 4x4 chroma blocks for ChromaArrayType equal to 3

This process is only invoked when ChromaArrayType is equal to 3.

The derivation process for neighbouring 4x4 chroma block in 4:4:4 chroma format is identical to the derivation process for neighbouring 4x4 luma block as specified in subclause 6.4.8.3 when substituting the term "luma" with the term "Cb" or the term "Cr", and substituting the term "luma4x4BlkIdx" with the term "cb4x4BlkIdx" or the term "cr4x4BlkIdx" in all places in subclause 6.4.8.3.

In subclause 7.3.2.1 "Sequence parameter set RBSP syntax", make the following changes.

*Replace the syntax element "***residual_colour_transform_flag***" with the syntax element "***separate_colour_plane_flag***".*

*Replace the expression "*profile_idc == 144*" with "*profile_idc == 44 || profile_idc == 244*".*

*Replace the expression "*i < 8*" with "*i < ((chroma_format_idc != 3) ? 8 : 12)*".*

*In subclause 7.3.2.2 "Picture parameter set RBSP syntax", replace the expression "*i < 6 + 2* transform_8x8_mode_flag*" with "*i < 6 + ((chroma_format_idc != 3) ? 2 : 6) * transform_8x8_mode_flag".

Replace the slice data partition B RBSP syntax table of subclause 7.3.2.9.2 "Slice data partition B RBSP syntax" with the following:

slice_data_partition_b_layer_rbsp( ) {	**C**	**Descriptor**
slice_id	All	ue(v)
if(separate_colour_plane_flag == 1)		
colour_plane_id	All	u(2)
if(redundant_pic_cnt_present_flag)		
redundant_pic_cnt	All	ue(v)
slice_data( ) /* only category 3 parts of slice_data( ) syntax */	3	
rbsp_slice_trailing_bits( )	3	
}		

Replace the slice data partition C RBSP syntax table of subclause 7.3.2.9.3 "Slice data partition C RBSP syntax" with the following:

slice_data_partition_c_layer_rbsp( ) {	**C**	**Descriptor**
slice_id	All	ue(v)
if(separate_colour_plane_flag = = 1)		
colour_plane_id	All	u(2)
if(redundant_pic_cnt_present_flag)		
redundant_pic_cnt	All	ue(v)
slice_data( ) /* only category 4 parts of slice_data( ) syntax */	4	
rbsp_slice_trailing_bits( )	4	
}		

Replace the slice header syntax table of subclause 7.3.3 "Slice header syntax" with the following:

slice_header( ) {	**C**	**Descriptor**
first_mb_in_slice	2	ue(v)
slice_type	2	ue(v)
pic_parameter_set_id	2	ue(v)
if(separate_colour_plane_flag = = 1)		
colour_plane_id	2	u(2)
frame_num	2	u(v)
if(!frame_mbs_only_flag) {		
field_pic_flag	2	u(1)
if(field_pic_flag)		
bottom_field_flag	2	u(1)
}		
if(nal_unit_type = = 5)		
idr_pic_id	2	ue(v)
if(pic_order_cnt_type = = 0) {		
pic_order_cnt_lsb	2	u(v)
if(pic_order_present_flag && !field_pic_flag)		
delta_pic_order_cnt_bottom	2	se(v)
}		
if(pic_order_cnt_type = = 1 && !delta_pic_order_always_zero_flag) {		
delta_pic_order_cnt[0 **]**	2	se(v)
if(pic_order_present_flag && !field_pic_flag)		
delta_pic_order_cnt[1 **]**	2	se(v)
}		
if(redundant_pic_cnt_present_flag)		
redundant_pic_cnt	2	ue(v)
if(slice_type = = B)		
direct_spatial_mv_pred_flag	2	u(1)
if(slice_type = = P | | slice_type = = SP | | slice_type = = B) {		
num_ref_idx_active_override_flag	2	u(1)
if(num_ref_idx_active_override_flag) {		

num_ref_idx_l0_active_minus1	2	ue(v)
if(slice_type = = B)		
num_ref_idx_l1_active_minus1	2	ue(v)
}		
}		
ref_pic_list_reordering( )	2	

if((weighted_pred_flag &&
    (slice_type = = P || slice_type = = SP)) ||
    (weighted_bipred_idc = = 1 && slice_type = = B))		

pred_weight_table( )	2	
if(nal_ref_idc != 0)		
dec_ref_pic_marking( )	2	

if(entropy_coding_mode_flag && slice_type != I &&
    slice_type != SI)		

cabac_init_idc	2	ue(v)
slice_qp_delta	2	se(v)
if(slice_type = = SP || slice_type = = SI) {		
if(slice_type = = SP)		
sp_for_switch_flag	2	u(1)
slice_qs_delta	2	se(v)
}		
if(deblocking_filter_control_present_flag) {		
disable_deblocking_filter_idc	2	ue(v)
if(disable_deblocking_filter_idc != 1) {		
slice_alpha_c0_offset_div2	2	se(v)
slice_beta_offset_div2	2	se(v)
}		
}		

if(num_slice_groups_minus1 > 0 &&
    slice_group_map_type >= 3 && slice_group_map_type <= 5)		

slice_group_change_cycle	2	u(v)
}		

In the syntax table of subclause 7.3.3.2 "Prediction weight table syntax", replace the syntax expression:

chroma_format_idc != 0

with:

ChromaArrayType != 0

(in all 3 occurrences)

In the syntax table of subclause 7.3.5.1 "Macroblock prediction syntax", replace the syntax expression

chroma_format_idc != 0

with:

ChromaArrayType == 1 || ChromaArrayType == 2

(in one occurrence)

Replace the content of subclause 7.3.5.3 "Residual data syntax" with the following:

residual( ) {	**C**	**Descriptor**
if(!entropy_coding_mode_flag)		
residual_block = residual_block_cavlc		
else		
residual_block = residual_block_cabac		
residual_luma(i16x16DClevel, i16x16AClevel, level, level8x8)		
Intra16x16DCLevel = i16x16DClevel		
Intra16x16ACLevel = i16x16AClevel		
LumaLevel = level		
LumaLevel8x8 = level8x8		
if(ChromaArrayType == 1 || ChromaArrayType == 2) {		
NumC8x8 = 4 / (SubWidthC * SubHeightC)		
for(iCbCr = 0; iCbCr < 2; iCbCr++)		
if(CodedBlockPatternChroma & 3) /* chroma DC residual present*/		
residual_block(ChromaDCLevel[iCbCr], 4 * NumC8x8)	3 | 4	
else		
for(i = 0; i < 4 * NumC8x8; i++)		
ChromaDCLevel[iCbCr][i] = 0		
for(iCbCr = 0; iCbCr < 2; iCbCr++)		
for(i8x8 = 0; i8x8 < NumC8x8; i8x8++)		
for(i4x4 = 0; i4x4 < 4; i4x4++)		
if(CodedBlockPatternChroma & 2) /* chroma AC residual present */		
residual_block(ChromaACLevel[iCbCr][i8x8*4+i4x4],15)	3 | 4	
else		
for(i = 0; i < 15; i++)		
ChromaACLevel[iCbCr][i8x8*4+i4x4][i] = 0		
} else if(ChromaArrayType == 3) {		
residual_luma(i16x16DClevel, i16x16AClevel, level, level8x8)		
CbIntra16x16DCLevel = i16x16DClevel		
CbIntra16x16ACLevel = i16x16AClevel		
CbLevel = level		
CbLevel8x8 = level8x8		
residual_luma(i16x16DClevel, i16x16AClevel, level, level8x8)		
CrIntra16x16DCLevel = i16x16DClevel		

CrIntra16x16ACLevel = i16x16AClevel		
CrLevel = level		
CrLevel8x8 = level8x8		
}		
}		

Replace the content of subclause 7.3.5.3.2 "Residual block CABAC syntax" with the following:

residual_block_cabac(coeffLevel, maxNumCoeff) {	**C**	**Descriptor**
if(maxNumCoeff != 64 || (ChromaArrayType = = 3))		
coded_block_flag	3 | 4	ae(v)
if(coded_block_flag) {		
numCoeff = maxNumCoeff		
i = 0		
do {		
significant_coeff_flag[i **]**	3 | 4	ae(v)
if(significant_coeff_flag[i]) {		
last_significant_coeff_flag[i **]**	3 | 4	ae(v)
if(last_significant_coeff_flag[i]) {		
numCoeff = i + 1		
for(j = numCoeff; j < maxNumCoeff; j++)		
coeffLevel[j] = 0		
}		
}		
i++		
} while(i < numCoeff - 1)		
coeff_abs_level_minus1[numCoeff - 1 **]**	3 | 4	ae(v)
coeff_sign_flag[numCoeff - 1 **]**	3 | 4	ae(v)
coeffLevel[numCoeff - 1] = (coeff_abs_level_minus1[numCoeff – 1] + 1) * (1 – 2 * coeff_sign_flag[numCoeff – 1])		
for(i = numCoeff - 2; i >= 0; i--)		
if(significant_coeff_flag[i]) {		
coeff_abs_level_minus1[i **]**	3 | 4	ae(v)
coeff_sign_flag[i **]**	3 | 4	ae(v)
coeffLevel[i] = (coeff_abs_level_minus1[i] + 1) * (1 – 2 * coeff_sign_flag[i])		
} else		
coeffLevel[i] = 0		
} else		
for(i = 0; i < maxNumCoeff; i++)		
coeffLevel[i] = 0		
}		

Insert a new subclause 7.3.5.3.3 as follows:

7.3.5.3.3 Residual luma data syntax

residual_luma(i16x16DClevel, i16x16AClevel, level, level8x8) {	**C**	**Descriptor**
if(!entropy_coding_mode_flag)		
residual_block = residual_block_cavlc		
else		
residual_block = residual_block_cabac		
if(MbPartPredMode(mb_type, 0) = = Intra_16x16)		
residual_block(i16x16DClevel, 16)	3	
for(i8x8 = 0; i8x8 < 4; i8x8++)		
if(!transform_size_8x8_flag | | !entropy_coding_mode_flag)		
for(i4x4 = 0; i4x4 < 4; i4x4++) {		
if(CodedBlockPatternLuma & (1 << i8x8))		
if(MbPartPredMode(mb_type, 0) = = Intra_16x16)		
residual_block(i16x16AClevel[i8x8*4+ i4x4], 15)	3	
else		
residual_block(level[i8x8 * 4 + i4x4], 16)	3 | 4	
else if(MbPartPredMode(mb_type, 0) = = Intra_16x16)		
for(i = 0; i < 15; i++)		
i16x16AClevel[i8x8 * 4 + i4x4][i] = 0		
else		
for(i = 0; i < 16; i++)		
level[i8x8 * 4 + i4x4][i] = 0		
if(!entropy_coding_mode_flag && transform_size_8x8_flag)		
for(i = 0; i < 16; i++)		
level8x8[i8x8][4 * i + i4x4] = level[i8x8 * 4 + i4x4][i]		
}		
else if(CodedBlockPatternLuma & (1 << i8x8))		
residual_block(level8x8[i8x8], 64)	3 | 4	
else		
for(i = 0; i < 64; i++)		
level8x8[i8x8][i] = 0		
}		

In subclause 7.4.1.2.5 "Order of VCL NAL units and association to coded pictures", replace the following paragraph:

- Otherwise (arbitrary slice order is not allowed), the order of coded slice of an IDR picture NAL units shall be in the order of increasing macroblock address for the first macroblock of each coded slice of an IDR picture NAL unit.

with:

- Otherwise (arbitrary slice order is not allowed), the following applies.
  - If separate_colour_plane_flag is equal to 0, the order of coded slices of IDR picture NAL units shall be in the order of increasing macroblock address for the first macroblock of each coded slice of an IDR picture NAL unit.